MODEL-BASED TESTING FOR LARGE BUSINESS INFORMATION SYSTEMS: A METHOD WITH SCRIPT BASED TEST TECHNOLOGIES

Khalid Ebanhesaten, Detlef Streitferdt, Franz-Felix Füssl and Jirka Ciemala

*Corresponding Author: Khalid Ebanhesaten khalid.ebanhesaten@metrosystems.net

The motivation for software testing is the quest for quality in software products. A robust and smoothly running software with solid interfaces is the main goal of developers. In the software engineering domain the term “testing” has different meanings, starting with accurate compiling for systems up to structured and methodic testing by using script based testing technologies and testmodels for E2E\textsuperscript{1}-business processes. Distributed BIS\textsuperscript{2} are characterized by heterogeneous systems and business processes, whose complexity is steadily growing (Mlynarski \textit{et al}., 2009). For such systems the communication through interfaces has significantly changed. Instead of testing isolated business processes, testing of distributed and cooperating business processes has become more important. As a result of this complexity combined together with the omnipresent competition and time pressure a test method needs to be designed, which is aligned to the process and know-how-owners with models as well as script based test automation strategies. This will notably reduce testing time and enable successive improvement of test coverage. This paper introduces Model Based Testing (MBT\textsuperscript{3}) with a tool chain and presents a novel method which implements the MBT of BIS using an extended BPMN\textsuperscript{4} as modelling notation.

**Keywords:** Model Based Testing, Business Information Systems, Business Process Modelling Notation, Software Testing

INTRODUCTION

During the past few years the complexity of extensions and changes in distributed BIS has constantly grown (Zeiss \textit{et al}., 2008). Companies are aiming at customer requirements of business processes as well as complex software
solutions. Those software solutions are used especially for B2B processes at a constantly growing complexity. On the one hand standardized business processes are extended and adapted to company requirements. On the other hand new business processes and functionalities are developed and implemented. This leads to dependencies between individual software systems in relation to the process what leads to a complex network.

The different and partially heterogeneous software systems require a complete E2E regression testing throughout all processes. Permanent changes of distributed BIS driven by the daily business require several test phases. The sum of the regression test cases for all software systems of a BIS increases the number of business-critical test cases dramatically. A complete test without a tool assisted method results in an unjustifiable effort. An approach to support regression testing on the method level as well as with the needed tools is MBT as described in Monalisa and Rajib (2010), Sokenou (2006), and Mlynarski (2010). This article is focused on the domain of the distributed BIS. The involved functional departments within a company and the process owners generally do not have the needed technical background/know-how. Thus, new modelling approaches for the test phase are required. BPMN is a modelling language (OMG1, 2011), which is more frequently used by the functional departments. Currently the usage of BPMN stops at the modeling level. This is a disadvantage compared to the UML which is used for various transformations to other notations including source code.

Our solution is the extension of the BPMN specification by the elements needed for test modelling. This article proposes an extension of the BPMN and a framework to use this extension. The modelling and generation phase are supported by the tool IBA5 and MBTSuite6. With this setup we achieve the creation of a manual test case out of a test model and its centralized storage in a test management tool. An additional step is needed towards a generated test script with reusable test steps based on the manual test case. Our solution to bridge this gap of several MBT approaches (Monalisa and Rajib, 2010), (Sokenou, 2006) is also discussed in this article.

STATE OF THE ART

Each large and complex software system necessarily contains errors. An error defines a discrepancy between the actual and the expected behavior of software. This discrepancy can occur as a numerical value or a missed time measurement which is needed for the execution of a process. Testing is meant to identify those discrepancies. Based on Pretschner and Philips, (2005) and Utting and Legeard (2007), in a software project 30-50% of the development efforts are used for error handling. An overview of the quantity and type of errors can be found by screening the error database of the test management tool in use. Testing does not guarantee perfection (Utting et al., 2006), but it primarily contributes to detect failures. A functional specification defines the correct behavior of the system being tested. Therefore this specification is of the utmost importance. Thus, the software test can only check the correct implementation

---

5 Innovator of Business Analyst is a Tool from MID GmbH.
6 Testcase Generator from Seppmed GmbH.
of requirements being integrated in the specification (software verification) (OMG1, 2011). It is crucial to define the expected results of the software and compare them with the actual results (so called “oracle problem”) (Bertolino, 2007). Since software with defects can still show the expected behavior for most of the input values, high quality test cases with a large potential of finding errors are crucial (Utting et al., 2006). This is why the excellence of a test is heavily depending on the selected and specified test cases.

But the definition of this excellence as well as the number of test cases show, that testing without support of testing tools is barely possible in an acceptable time- as well as resource effort. Especially the excellence of a test case requires huge empirical values from functional departments, which can define critical business processes and give valuable advice during the test case creation. The main question is how to manage the amount of test cases in a regression test while having time and cost pressure. One approach of the authors is the MBT on the basis of BPMN. MBT in general is very diverse (Sokenou, 2006) which is why several approaches within MBT can be found in research.

Many researches including Monalisa and Rajib (2010), Sokenou (2006), Mlynarski (2010), and Mlynarski et al. (2009) come to the conclusion that testing with models generally result in an improvement of the test process. Those theoretic approaches also show generation of mature test scripts based on UML test models. Those generated test cases respectively test scripts are executed on small self-contained components especially in the domain of embedded software systems appearing in control units of cars, medical devices or aerospace technology. These embedded software systems are based on very complex algorithms and require a high degree of technical know-how. Target groups are technical engineers, which are capable not just to implement, but also to interpret and check for discrepancies.

The first steps of testing distributed BIS on the basis of models are shown in Wieczorek and Stefanescu (2010) and Mlynarski (2010). While the above mentioned tasks are addressing the generation of test cases from UML state diagrams, UML sequence diagrams or use case diagrams, Jääskeläinen et al. (2009) is based on creation of test scenarios from UML activity diagrams. Within the scope of the thesis Dias et al. (2007) a theoretical approach has been presented, which has never been executed in life praxis before and thus appears rather vague. UML is a very powerful modelling notation and rather deterrent for functional departments and its employees like financial accountants or generally accounting experts. BPMN provides quite some advantages (OMG1, 2011) to ease the work of functional departments. Hence, the establishment of a MBT approach on basis of BPMN models is an encouraging attempt to enjoy greater acceptance among functional departments. First theoretical approaches to model test cases on basis of BPMN were shown in Wieczorek and Stefanescu (2010) and Mlynarski (2010). However these two reports provided only examples without showcasing or outline the technical realization. The survey in Götz et al. (2009) demonstrated and assessed many MBT testing tools, nevertheless all tools revealed no support of BPMN, which is going to be executed now. On the basis of IBA as modelling tool, MBTSuite as test case generator, HPQC as test management
tool and QTP as test automation tool a method is to be drafted, which is realizing an MBT approach for distributed BIS on Basis of BPMN. Figure 1 depicts the overview of the solution approach.

**CHALLENGES – TECHNICAL REALIZATION**

Challenges within the MBT approach for distributed BIS are based on a variety of different software systems involved in the process and their respective interfaces. Some theoretical approaches within the domain of distributed BIS with regards to MBT are included in Sokenou (2006) and Wieczorek and Stefanescu (2010). This paper addresses the big challenges of using a BPMN test notation to model and refine test cases for the corresponding requirements of a specification. At the same time test scripts should be generated out of the new BPMN notation to execute the test cases in an automated manner.

Figure 2 shows a summary of this approach.

**General Approach**

The starting point is a functional specification (requirements) which is fundamental for the test case specification. To guarantee traceability from requirements via test cases to the defect management it is recommended to choose a software solution which is supporting test management as well as requirements management. HP ALM (Hewlett Packard Application Lifecycle Management) supports both test case management as well as the requirement engineering phase. Any requirement might now be created and directly allocated to a test case that was generated out of the test model.

In addition, HP ALM integrates MBTSuite\(^7\) and IBA\(^8\). Thus, requirements can be transferred from HPQC\(^9\) to IBA while keeping traces. Furthermore the test cases generated in MBTSuite can also be traced to requirements, ensuring the complete

---

\(^7\) MBTSuite: Model based Testing Suite

\(^8\) IBA tool, produced by MID GmbH

\(^9\) Hewlett Packard Quality Center is a test management tool as a part of HP ALM

---

This article can be downloaded from http://www.ijerst.com/currentissue.php

42
traceability from test cases to the test model and vice versa.

IBA was not designed to model test cases. Here, adaptations of the meta-model and for the BPMN specification have been developed. Now the test cases can be created and customized with a new test notation.

Within the scope of this paper IBA has been...
extended with test elements and test specifications. These extensions enable to model business processes as well as visualize all paths for such processes. Finally, a test case generator transforms the test cases being specified in BPMN models into executable test scripts.

For the proposed testing solution for distributed BIS, all our tools need to cooperate in a network. Figure 3 outlines the procedure (the sequence is numbered in the figure) to generate a test case/test script starting with requirements related to the test model. The implementation of test scripts is realized in functional libraries, ensuring a gradual enhancement of a test pool and thus, regression testing. The test model is built for regression testing and ensures the synchronization of the test case core with the test case specification.

Changes of the test model addressing the continuous requirements changes will be generated by MBTSuite as delta test cases and added to the portfolio of test scripts. Since IBA supports modelling with UML, test data can be directly included into the UML class diagrams and individually bound to test steps. Via the conversion of such test data into excel tables the subsequent test automation was done with HP Quick Test Professional (QTP).

MBTSuite allows the generation of test cases from BPMN test models. The generation follows certain rules and coverage criteria, which can be defined by the tester. MBTSuite is responsible for placing the generated test cases from BPMN test models into the predefined file structure of the test management tool. Besides the file structure, a vital naming convention for the test case names of the test model has been developed.

The naming convention for the test case names, as shown in Figure 4, has three parts. First, the name of the Softwaresystem identifies the software component of the complete business system. Here, we recommend an abbreviation of this name, like “sap” in the example. Second, the Functionalmodule is the name of the function implemented in the Softwaresystem. The third part of the name is the test case itself (Testcasename).

These naming conventions have to be enforced throughout modeling and implementation of functions developed by different development teams of a company distributed throughout different locations in the world.

QTP from HP is the tool for scripting the test step logic as well as the tool for the automated executing of the test scripts. For the fully automated use of the tool chain the interaction between the individual tools requires further extensions.

**Extension of BPMN Specification by Test Elements**

As a first step, BPMN has to be extended by test elements to allow modeling and specifying

---

**Figure 4: Naming Conventions**

```
<<Softwaresystem_Functionalmodule_Testcasename>>
```

**Example:** sap_login_simplelogin

This article can be downloaded from http://www.ijerst.com/currentissue.php
activities as test steps, by means of stereotypes. BPMN distinguishes between its meta model and the notation. An extract of the extension of the class model and respectively of the specification can be found in Figure 5.

The proposed extension is based on the BPMN meta model (OMG1 2011). The new class “TestElement” has been added, in conformance to the BPMN specification (OMG1, 2011) and (OMG2, 2011).

The extension for the BPMN specification implies the extension of the IBA specification. The fields “TCName4”, “TCDescription”, “Prio”, and “Duration” have been added to capture test specific characteristics in the model. Thus, the IBA policy of modelling has been extended by these fields. With this extension it is possible to model individual test steps with pre- and post-conditions. In Figure 6 the textual notation is shown on the left the corresponding test model is shown on the right part of the figure.
As shown, the realized extension can be used with the configuration editor of IBA. For the extension of the meta model, the file “innovator.aob” in the installation directory of the Innovator tool was extended accordingly (after being copied to “B2B.aob”). Now, this new file holds the profile including all enhancements for our new test notation.

Extension of HP QC by Adaptation Functionality

HPQC has been extended by a field “Test Name” which also holds the implementation logic. The name for this field is generated by the test case generator. The implementation logic checks whether the test process logic within the abstract syntax tree exists and fits to the generated test script, shown in Figure 7.

The functions “GetTestName” and “Find_Function_in_Functionalibrary” ensure the access to the abstract syntaxtree of QTP and enable searching for a corresponding function based on the test case name, which was handed over in test script. In addition the functions “ask” whether the identified function contains implementation logic for a test case (an example is shown in Figure 8). For such cases an
automatic referencing is supported by the sub-
function "keepAttInTestRun". In case a function
name is not found in the library a notification to
implement a function with test logic to this specific
test script name will be issued to the developer.
Both searching the syntax tree as well as
referencing the test script function are taking
place automatically.

**Embedding the Functional Libraries and Test Data**

Test cases should be prioritized by risk and
criticality in order to consider all business critical
test cases on each test run. The combination of
UML class diagrams and with simple tables turned
out to be the most efficient way to create test
cases. Here, the specification of test data is very
important together with test scripts for verification.

The example in Figure 9 demonstrates how
to create a test step in the model out of a UML
class diagram with attributes and parameters. At
the same time it shows how test data can either
be converted to spreadsheet tables or created in
a spreadsheet tool. The conversion of the UML
class diagram to, e.g. Excel can be done directly
from the IBA tool. The verification of the test data
is also reflected within the test model as "<<VP>>
"Verification Point". Here, the transmitted test data
is checked for consistency. Only if this verification
passed the test case will be handed over to the
test execution. In case of problems error
messages automatically point out the missing
data to the developer.

**Usage of the Method**

The method presented in this paper uses several
tools, which are HPQC, QTP, IBA and MBTSuite.
With these tools together with the extensions,
testing of end-to-end business processes of
distributed BIS is now possible. The six steps of
this method are shown in Figure 10. It is important
to mention that all business processes have
dependencies to several software systems which
can also be "model tested". This method is based
on a combination of model- and script-based
testing technologies which enable the automation
of test cases.

The test execution is performed by a test
management tool. The task of this test
management tools is to administer the generated
test cases and test scripts and to document the
results. Additionally a test management tool provides a central storage of required test data for the tests. For that purpose we defined a fixed file structure enable a very easy access to the test scripts.

This method can be applied to all distributed BIS that are based on business processes. For the concrete implementation it is important to adhere to the naming conventions and to the correct definition of the individual file structures of the test data.

Finally, the generated and adapted test scripts are transferred directly into the test lab and are available for execution. A test scheduler is taking over the coordinated test execution and is also responsible for writing the test results in a test assessment sheet.

**SUMMARY AND CONCLUSION**

Within the test case specification, which is the base for test case modelling, the responsible functional department needs to be involved. Thereby, uncertainties or errors within the models can be identified and resolved immediately. As a benefit, bugs can be localized and fixed at an early stage, leading to significantly lower failure costs in productive systems.

The subject of this paper is a new method to generate test cases on the basis of BPWN test models with an extended BPMN notation and an automated test cases execution. This approach bridges the gaps of already existing MBT approaches on basis of UML. As example we have demonstrated the process of a login procedure from the model to the test script generation. Furthermore the execution of the generated test scripts by a test scheduler has been addressed.

With this method our evaluation showed that test coverage can be increased gradually and bugs in productive systems can be reduced. The evaluation was only one part of the study, another evaluation needs to prove the sustainability of this method in the future.

**REFERENCES**


